This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Myun-Hoon SUNWOO	Examiner: Not Yet Assigned
Serial No.:	10/608,511	Group Art Unit: 2631
Filed:	June 27, 2003	Docket: 678-1202
For:	**MODULATION APPARATUS USING MIXED-RADIX FAST FOURIER TRANSFORM**	Dated: May 10, 2004

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF PRIORITY DOCUMENT

Sir:

Enclosed is a certified copy of Korean Appln. No. 2002-36216 filed on June 27, 2002, from which priority is claimed under 35 U.S.C. §119.

Respectfully submitted,

Paul J. Farrell
Registration No. 33,494
Attorney for Applicant

DILWORTH & BARRESE, LLP
333 Earle Ovington Boulevard
Uniondale, New York 11553
(516) 228-8484

CERTIFICATE OF MAILING UNDER 37 C.F.R. § 1.8 (a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail, postpaid in an envelope, addressed to the: Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on May 10, 2004.

Dated: May 10, 2004

Paul J. Farrell

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : Application Number	10-2002-0036216
출 원 년 월 일 : Date of Application	2002년 06월 27일 JUN 27, 2002
출 원 인 : Applicant(s)	삼성전자주식회사 외 1명 SAMSUNG ELECTRONICS CO., LTD., et al.

2003 년 08 월 01 일

특 허 청

COMMISSIONER

【서지사항】

【서류명】	출원인 변경 신고서
【수신처】	특허청장
【제출일자】	2003.06.27
【구명의인】	
【명칭】	학교법인대우학원
【출원인코드】	2-1999-901351-3
【사건과의 관계】	출원인
【신명의인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이건주
【대리인코드】	9-1998-000339-8
【포괄위임등록번호】	2003-001449-1
【사건의 표시】	
【출원번호】	10-2002-0036216
【출원일자】	2002.06.27
【발명의 명칭】	연속 처리 구조를 갖는 인-프레이스 알고리즘 혼합-기수 에프에프티 프로세서
【변경원인】	일부양도
【취지】	특허법 제38조제4항·실용신안법 제20조·의장법 제24조 및 상표법 제12조 제1항의 규정에 의하여 위와 같이 신고합니다. 대리인 이건주 (인)
【수수료】	13,000 원
【첨부서류】	1. 양도증_1통 2.인감증명서_1통 3.위임장_1통

【서지사항】

【서류명】 출원인 변경 신고서
【수신처】 특허청장
【제출일자】 2003.05.24
【구명의인】
【성명】 선우명훈
【출원인코드】 4-1998-035863-9
【사건과의 관계】 출원인
【신명의인】
【명칭】 학교법인 대우학원
【출원인코드】 2-1999-901351-3
【대리인】
【성명】 오세중
【대리인코드】 9-1998-000467-5
【포괄위임등록번호】 2003-025047-8
【사건의 표시】
【출원번호】 10-2001-0007555
【출원일자】 2001.02.15
【심사청구일자】 2001.02.15
【발명의 명칭】 피알엠엘 하드디스크 드라이브용 에프아이알 필터
【사건의 표시】
【출원번호】 10-2001-0022427
【출원일자】 2001.04.25
【심사청구일자】 2001.04.25
【발명의 명칭】 리드 -솔로몬 부호화 및 복호화를 위한 프로그래머블 프로세서의 유한체 연산기 회로 및 연산방법
【사건의 표시】
【출원번호】 10-2001-0043711
【출원일자】 2001.07.20
【심사청구일자】 2001.07.20
【발명의 명칭】 디에스피 프로세서 및 마이크로프로세서의 실시간 영상데이터 처리를 위한 연산회로 및 그 연산방법
【사건의 표시】
【출원번호】 10-2001-0043712

【출원일자】 2001.07.20
【심사청구일자】 2001.07.20
【발명의 명칭】 프로그래머블 프로세서에서의 비터비 디코딩 연산 방법 및 그 연산을 실행하기 위한 연산회로

【사건의 표시】
【출원번호】 10-2001-0043713
【출원일자】 2001.07.20
【심사청구일자】 2001.07.20
【발명의 명칭】 프로그래머블 프로세서에서 고속에프에프티 연산을 위한 에프에프티 연산방법 및 그 연산방법을 실행하기 위한 에프에프티 연산회로

【사건의 표시】
【출원번호】 10-2001-0047292
【출원일자】 2001.08.06
【심사청구일자】 2001.08.06
【발명의 명칭】 리드 -솔로몬 복호기의 고속 수정 유클리드 알고리즘 연산방법 및 연산회로

【사건의 표시】
【출원번호】 10-2002-0036214
【출원일자】 2002.06.27
【심사청구일자】 2002.06.27
【발명의 명칭】 에러궤환을 이용한 블라인드 적응결정 궤환 등화기

【사건의 표시】
【출원번호】 10-2002-0036216
【출원일자】 2002.06.27
【심사청구일자】 2002.06.27
【발명의 명칭】 연속 처리 구조를 갖는 인-프레이스 알고리즘 혼합 기수 에프에프티 프로세서

【사건의 표시】
【출원번호】 10-2002-0069823
【출원일자】 2002.11.11
【발명의 명칭】 씨디엠에이 통신 시스템의 확산 및 변조기

【사건의 표시】
【출원번호】 10-2002-0078393
【출원일자】 2002.12.10

【심사청구일자】	2002.12.10
【발명의 명칭】	프로그래머블 프로세서에서의 고속 푸리에 변환 연산회로 및 연산방법
【변경원인】	전부양도
【취지】	특허법 제38조제4항·실용신안법 제20조·의장법 제24조 및 상표법 제12조 제1항의 규정에 의하여 위와 같이 신고합니다. 대리인 오세중 (인)
【수수료】	130,000 원
【첨부서류】	1. 양도증_10통 2.인감증명서[양도인]_1통 3.위임장[양도인, 양수인]_1통

【서지사항】

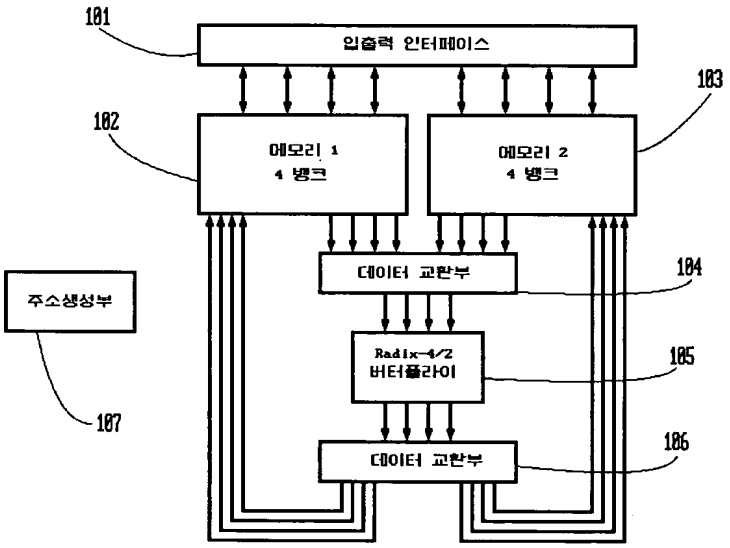
【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【제출일자】	2002.06.27
【발명의 명칭】	연속 처리 구조를 갖는 인-프레이스 알고리즘 혼합-기수 에프에프티 프로세서
【발명의 영문명칭】	Continuous Flow Mixed-radix FFT Processor with an In-place Algorithm
【출원인】	
【성명】	선우명훈
【출원인코드】	4-1998-035863-9
【대리인】	
【성명】	오세중
【대리인코드】	9-1998-000467-5
【발명자】	
【성명】	선우명훈
【출원인코드】	4-1998-035863-9
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인 오세중 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	9 면 9,000 원
【우선권주장료】	0 건 0 원
【심사청구료】	4 항 237,000 원
【합계】	275,000 원
【감면사유】	개인 (70%감면)
【감면후 수수료】	82,500 원
【첨부서류】	1. 요약서·명세서(도면)_1통 2.위임장_1통

【요약서】

【요약】

본 발명은 OFDM(Orthogonal Frequency Division Multiplexing), DMT(Discrete MultiTone) 모뎀의 핵심 기능부에 해당하는 FFT(Fast Fourier Transform) 프로세서에 관한 것으로, 특히 상기 FFT 프로세서는 혼합-기수(Mixed-radix)에 다중 뱅크 메모리를 위한 인-프레이스(In-place) 알고리즘을 적용하여 순차적 입력과 출력을 동시에 수행함으로써 4개의 뱅크로 구성된 2N워드 메모리만으로 연속처리를 구현하여 고속 연산과 동시에 메모리 복잡도를 최소화한 FFT 프로세서를 제공한다.

【대표도】



【색인어】

FFT 프로세서

【명세서】

【발명의 명칭】

연속 처리 구조를 갖는 인-프레이스 알고리즘 혼합-기수 에프에프티 프로세서{Continuous Flow Mixed-radix FFT Processor with an In-place Algorithm}

【도면의 간단한 설명】

도1은 종래의 다중 뱅크 구조를 사용하지 않는 혼합-기수(Mixed-radix) 알고리즘 FFT 프로세서의 구조를 나타낸 블록도.

도2는 종래의 인-프레이스(In-place) 알고리즘을 사용하지 않는 혼합-기수 알고리즘 FFT 프로세서의 구조를 나타낸 블록도.

도3은 종래의 다중 뱅크 메모리를 위한 기수-4(이하; Radix-4) 인-프레이스 알고리즘을 나타낸 도면.

도4는 본 발명에 따른 연속 처리 구조를 갖는 In-place 알고리즘 혼합-기수FFT 프로세서의 구조를 나타낸 블록도.

도5는 본 발명에 따른 32-포인트 혼합-기수 FFT 프로세서의 연산 흐름도.

도6a는 도4의 FFT 프로세서에 사용되는 Radix-4/Radix-2 버터플라이 회로의 블록도.

도6b는 도6a의 Radix-2 버터플라이 연산 시 등가 버터플라이 쌍을 나타낸 도면.

도7은 도3의 Radix-4 알고리즘만을 사용할 경우에 연속 처리를 구현하기 위한 FFT 연산 흐름도.

도8a는 4^n-포인트 FFT 연산을 위해 Radix-4 알고리즘만을 사용할 경우에 출력의 디지트 리버스(Reverse) 순서를 나타낸 도면.

도8b는2^n(n=3, 5, 7, 9, …)- 포인트 FFT 연산을 위해 Radix-2를 함께 사용하는 혼합-기수 알고리즘을 사용할 경우에 출력의 비대칭적 리버스 순서를 나타낸 도면.

도9는 32-포인트 혼합-기수 FFT 연산인 도5의 열4에 해당하는 비대칭적 리버스 출력 순서를 나타낸 도면.

도10은 32-포인트 혼합-기수 FFT 연산인 도5의 열3에 해당하는 대칭적 리버스 출력 순서를 나타낸 도면.

도11은 2^n(n=3, 5, 7, 9, …)-포인트 FFT 연산을 위한 혼합-기수의 대칭적 리버스 출력 순서를 나타낸 도면.

도12는 도5의 열3에 해당하는 대칭적 리버스 출력 순서를 생성하기 위한 데이터 교환을 나타낸 흐름도.

도13은 32-포인트 혼합-기수 FFT 연산인 도5의 뱅크 인덱스 생성 방법을 나타낸 도면.

도14는 2^n(n=3, 5, 7, 9, …)-포인트 FFT 연산을 위한 혼합-기수의 뱅크 인덱스 생성 방법을 나타낸 도면.

<도면의 주요부분에 대한 부호의 설명>

10 : 버터플라이 연산부 11 : 입출력 인터페이스

12 : 메모리 제어기 13 : 듀얼포트 메모리

14 : 버터플라이 15 : 복소 곱셈기

16 : 회전인자 LUT 17 : 선택적 버터플라이

18,23,24 : MUX 21,22: 입력 RAM

25 : 공유 연산부 26,27 : Radix-2 연산부

28,29 : 출력 RAM 101 : 입출력 인터페이스

102, 103 : 4개의 뱅크로 구성된 N 워드 메모리

104 : 제1데이터 교환부

105 : Radix-4/Radix-2버터플라이 연산부

106 : 제2데이터 교환부 107 : 주소 생성부

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<29> 본 발명은 OFDM, DMT 모뎀의 핵심 기능부에 해당하는 FFT 프로세서에 관한 것으로, 특히 고속 연산을 위해 혼합-기수(Mixed-radix) 알고리즘과 다중 뱅크 메모리를 사용하고 메모리 복잡도의 최소화를 위해 버터플라이 입력을 읽어온 메모리 위치에 버터플라이 출력을 저장하는 인-플레이스(In-place) 알고리즘을 사용하며 순차적 입력과 출력을 동시에 수행함으로써 FFT 길이 N의 두 배인 2N 워드의 메모리만으로 연속처리가 가능하도록 하는 FFT 프로세서에 관한 것이다.

<30> OFDM 전송 방식은 다중 경로 채널에서의 고속 데이터 전송을 위해 제안되었으며, 종래의 단일 캐리어 전송 방식이 데이터를 직렬로 전송하여 각 심볼이 전 채널의 주파수 대역을 사용

하는 데 비해, OFDM, DMT 변복조의 입력 데이터를 부반송파의 수 만큼 직·병렬 변환하여 각각에 대응되는 부반송파(subcarrier)로 변조하는 방식이다. 이러한 부반송파를 이용한 변조는 DFT(Discrete Fourier Transform)을 이용하여 구현하며, 실제 하드웨어 설계에는 DFT나 IDFT(Inverse Discrete Fourier Transform)를 사용하지 않고 연산량을 줄이기 위해 FFT 알고리즘을 이용하였다. FFT 프로세서는 OFDM 시스템에 있어 가장 큰 복잡도를 가지며 고속 연산이 요구되어 구현이 까다로운 부분이다.

<31> 이러한 고성능을 요구하는 분야에는 파이프라인 구조 FFT 프로세서가 주로 사용되나 이 구조는 스테이지 수만큼의 연산부를 요구하여 포인트 수가 증가할 경우 많은 면적을 소모한다. 이러한 단점을 극복하고 적은 하드웨어 크기를 유지하기 위해 메모리 구조와 단일 버터플라이 연산부를 사용하는 프로세서들이 발표되고 있다.

<32> 그 한 예로서 기수-2(이하; Radix-2라 함) FFT 알고리즘을 사용한 메모리 구조 FFT 프로세서가 있다. 메모리 구조에 상기 Radix-2 알고리즘을 사용할 경우 곱셈기의 수를 최소화할 수 있어 적은 면적을 소모하는 FFT 프로세서의 구현이 가능하다.

<33> 그러나 상기 Radix-2 알고리즘을 사용한 메모리 구조 FFT 프로세서는 매우 많은 연산 싸이클을 소모하여 고속 연산이 요구되는 OFDM, DMT 시스템에 적합하지 않으며 요구되는 연산 시간을 만족시키기 위해서는 매우 높은 동작 주파수가 요구된다.

<34> 상기 메모리 구조를 갖는 FFT 프로세서의 다른 실시 예로서 도1은 암피온(AMPHION) 사에서 발표한 기수-4(이하; Radix-4라 함) 알고리즘을 기반으로 한 FFT 프로세서를 나타낸 것이다. 상기 Radix-4 알고리즘의 경우 Radix-2 알고리즘에 비해 1/2의 스테이지 수를 요구하

고 스테이지 당 버터플라이 연산 횟수도 Radix-2에 비해 1/2이기 때문에 훨씬 적은 버터플라이 연산 횟수를 가지며 이는 표1과 같다.

<35> 그러나 Radix-4 알고리즘은 4n의 길이를 갖는 FFT 연산만이 가능하므로 2n의 길이를 갖는 모든 FFT 연산을 가능하도록 하기 위해 Radix-4 알고리즘을 다른 Radix와 함께 사용하는 Mixed-radix 알고리즘이 요구된다. 표1의 마지막 열에 Radix-4, Radix-2의 Mixed-radix 알고리즘을 사용하였을 경우의 버터플라이 연산 횟수를 나타내었다.

<36> 【표 1】

Radix-2, Radix-4, 혼합-기수 알고리즘의 버터플라이 연산 회수

FFT 길이	Radix-2	Radix-4	혼합-기수
256	1,024	256	-
512	2,304	-	640
1,024	5,120	1,280	-
2,048	11,264	-	3,072
4,096	24,576	6,144	-
8,192	53,248	-	14,336

<37> 상기 혼합-기수 알고리즘을 사용하는 상기 도1의 FFT 프로세서는 Radix-4 버터플라이와 Radix-4/Radix-2 선택적 버터플라이 구조의 연동을 통해 Radix-4/Radix-8/Radix-16의 혼합-기수 연산을 수행한다.

<38> 상기 도1의 FFT 프로세서는 FFT 연산을 위한 입력 데이터를 프로세서 외부로부터 입력하고 FFT 연산이 수행된 결과 데이터를 프로세서 외부로 출력하며 FFT 연산 과정을 제어하는 입출력 인터페이스 및 제어기(11)와, 메모리 주소생성과 읽기/쓰기를 제어하는 메모리 제어기(12)와, 외부 입력 데이터 및 FFT 연산 중간의 데이터, 결과 데이터를 저장하는 1024-워드 듀얼 포트 메모리(13)와, Radix-4 버터플라이 연산중 덧셈과 뺄셈들을 수행하는 Radix-4 버터플

라이(14)와, 회전인자를 저장하는 회전인자 LUT(16)와, Radix-4 버터플라이 연산중 곱셈을 수행하는 복소 곱셈기(15)와, FFT 연산의 마지막 스테이지에서 Radix-2 혹은 Radix-4 버터플라이를 Radix-4 버터플라이 연산부(10)의 Radix-4 버터플라이 연산과 연동하여 Radix-8 혹은 Radix-16 연산을 수행 가능토록 하는 Radix-4/Radix-2 선택적 버터플라이(17)와, 마지막 스테이지에서만 선택적 버터플라이(17)를 사용하고 나머지 스테이지에서는 Radix-4 버터플라이 연산부(10)만을 사용하기 위한 MUX(18)로 구성된다. 상기 Radix-4 알고리즘의 경우 4개의 입력과 출력을 갖는 버터플라이로 구현된다. 4개의 입출력이 한 싸이클에 수행되어야 연산 싸이클을 최소화할 수 있으며 이를 위해 메모리를 다중 뱅크로 나누어 사용하여야 한다. 그러나, 상기 도1의 FFT 프로세서는 다중 뱅크 구조를 사용하지 않아 많은 연산 싸이클이 요구되어 Radix-4 연산의 장점을 살리지 못하고 있다.

<39> 상기 메모리 구조를 갖는 FFT 프로세서의 또 다른 실시예로써 도2는 드레이 엔터프라이즈(Drey Enterprise)사에서 발표한 혼합-기수 알고리즘과 다중 뱅크 구조를 갖는 FFT 프로세서를 나타낸 것으로, 하나의 RAM이 외부 입력 데이터를 저장하는 동안 나머지 RAM은 FFT 연산에 사용되는 두 개의 입력 RAM(21,22)과, 입력 RAM으로부터 버터플라이 입력을 받을지 출력 RAM으로부터 버터플라이 입력을 받을지를 결정하는 MUX(23)와, Radix-2 연산을 수행하는 스테이지에서 Radix-2 연산부들(26,27)로부터 Radix-2 버터플라이 출력을 받아 RAM에 저장하기 위해 사용되는 MUX(24)와, Radix-4 연산을 수행하는 스테이지에서는 Radix-4 연산을 수행하고 Radix-2 연산을 수행하는 스테이지에서는 Radix-2 연산을 수행하는 Radix-2/Radix-4 공유연산부(25)와, Radix-2 연산 스테이지에서 사용되는 Radix-2 연산부(26,27)와, 하나의 RAM이 FFT 연산에 사용되는 동안 나머지 RAM은 FFT 연산의 결과 데이터를 외부로 출력하는데 사용되는 출력

RAM(28,29)으로 구성된다. 본 구조는 Radix-4와 Radix-2의 혼합-기수 알고리즘을 사용하며 메모리도 다중 뱅크 구조를 사용한다. 다중 뱅크 구조의 사용으로 인해 연산 클럭 싸이클이 최소화된 구조이다.

<40> 그러나, 이 구조는 버터플라이 입력을 억세스한 메모리 위치에 버터플라이 출력을 저장하는 인-플레이스 알고리즘을 적용하지 않음으로 인해 FFT 연산에 N워드를 갖는 메모리 2개를 사용하는 구조이다. 상기 FFT 연산만을 위해서는 각각 4개의 뱅크로 구성된 메모리 2개를 사용하나 연속 처리를 위하여 입력과 출력을 위한 2개의 메모리를 더 사용한 구조이다. 따라서, 총4개의 메모리가 사용된다. 상기와 같이 인-플레이스 알고리즘을 적용하지 않고 입력과 출력을 하나의 메모리를 통해 동시에 수행하는 구조를 사용하지 않으면 메모리의 복잡도가 매우 커지는 문제점이 있다. 메모리의 경우 FFT 프로세서에서 대부분의 면적을 차지하는 블록 중에 하나이므로 상기 구조는 적은 하드웨어 복잡도의 장점을 살리지 못하고 있다.

<41> 상기 메모리 구조들의 메모리 복잡도를 최소화하기 위해 엘.쥐. 존슨(L. G. Johnson)이 발표한 인-플레이스 알고리즘의 16-포인트 FFT에 대한 실시예는 도3과 같다. 상기 인-플레이스 알고리즘은 메모리를 다중 뱅크로 나누어 사용할 경우를 위한 알고리즘이다. Radix-4 버터플라이 연산을 위해 4개의 데이터를 동시에 억세스하여야 하고, 억세스한 위치에 버터플라이 결과 4개를 동시에 저장하여야 한다. 이를 위해 메인 메모리를 4개의 뱅크(뱅크0, 뱅크1, 뱅크2, 뱅크3)로 나누어 사용하며 동시에 한 뱅크에서 여러개의 데이터를 억세스하지 않기 위해 적절한 어드레싱을 수행한다. 도3은 16-포인트 FFT에 대한 인-플레이스 메모리 어드레싱의 예이며 4개의 입력을 각기 다른 뱅크들로부터 읽어 오는 것이 가능하다. ①번 버터플라이의 경우 4개의 입력을 뱅크의 0번지, 뱅크1의 1번지, 뱅크2의 2번지, 뱅크3의 3번지로부터 읽어오고 버터플라이 연산 결과를 같은 뱅크들과 번지들에 저장한다. ②번 버터플라이의 경우 4개의 입력을

뱅크1의 0번지, 뱅크2의 1번지, 뱅크3의2번지, 뱅크0의 3번지로부터 읽어오고 버터플라이 연산 결과를 같은 위치에 저장한다. 도3에서 어느 뱅크를 사용하는지를 나타내는 뱅크 인덱스(뱅크 i)는 데이터 입력 카운트비트들을 2비트 디지트들로 나누어 모듈로-4 덧셈을 취함으로써 쉽게 구할 수 있다. 도3이 16-포인트 FFT이므로 16개의 데이터를 카운트하기 위해서 4 비트 카운터가 사용된다. 4 비트를 상위 2 비트와 하위 2 비트의 디지트로 나누어 모듈로-4 덧셈을 수행하는 방식으로 뱅크 인덱스를 구한다.

<42> 그러나, 상기 인-플레이스 알고리즘은 혼합-기수가 아니 고정된 기수에만 사용하기 위해 제안되었다.

<43> 다음은 연속 처리 구조를 갖는 종래의 구조에 대해 기술한다. 상기 메모리 구조에서 입력과 출력을 동시에 수행함으로써 N 워드의 크기를 갖는 메모리 2개만을 사용하여 연속처리가 가능한 메모리 구조의 FFT 프로세서가 알.래드후앤(R. Radhouane)에 의해 제안되었다. 상기 구조는 Radix-2알고리즘에서 DIF 연산을 할 경우와 DIT 연산을 할 경우에 각각 출력과 입력이 비트 리버스 특성을 갖는 것을 이용하여 한번은 DIF 연산을 수행하고 다음은 DIT 연산을 수행하는 방식으로 연속 처리를 구현하였다. 본 구조의 연산 방식을 표2에 나타내었다.

<44> 【표 2】

메모리 2개를 이용한 연속 처리

	메모리 1		메모리 2	
OFDM 심볼	메모리 상태	FFT-I/O 모드	메모리 상태	FFT-I/O 모드
0	C	DIF	I/O	NAT
1	I/O	BR	C	DIF
2	C	DIT	I/O	BR
3	I/O	NAT	C	DIT

<45> 상기 표2에서 OFDM 심볼은 FFT 연산의 길이에 해당하는 데이터를 의미한다. 예를 들어 256-포인트 FFT 연산의 경우 하나의 OFDM 심볼은 256개의 데이터들을 의미한다. C는 FFT 연산을 의미하고, I/O는 입출력을 수행함을 의미한다. NAT는 원래의 0, 1, 2, 3, …, N-1번지의 올바른 순서로 메모리 어드레싱을 수행하여 입출력하는 것을 의미하며, BR은 비트 리버스 어드레싱으로 메모리 입출력을 수행함을 의미한다. 표2의 0번 OFDM 심볼에서 메모리1이 DIF로 연산을 수행하는 동안 메모리2는 NAT 즉, 올바른 순서로 어드레싱을 수행하여 입출력을 수행한다. 다음 1번 OFDM 심볼에서 메모리1은 BR 즉, 비트 리버스어드레싱으로 입출력을 수행하고 메모리2는 DIF로 연산을 수행한다. 2번 심볼에서는 메모리1이 DIT로 연산을 수행하고 메모리2는 BR 즉, 비트 리버스 어드레싱으로 입출력을 수행한다. 다음 3번 심볼에서는 메모리1이 NAT 즉, 올바른 순서로 입출력을 수행하고 메모리2는 DIT로 연산을 수행한다. 다음 4번 심볼부터는 0, 1, 2, 3번 심볼의 연산흐름이 반복된다. 2개의 메모리로 연속처리가 가능하려면 하나의 메모리가 연산을 수행하는 동안 나머지 메모리는 순차적 순서를 갖는 입력과 출력을 동시에 수행 가능하여야 한다. 이와 같이 상기 구조에서는 두 메모리가 입출력과 FFT 연산을 번갈아 가며 수행하는 방식으로 2개의 메모리만으로 연속 처리가 가능하였다.

<46> 알카텔(Alcatel)사에서 발표한 종래의 구조가 3개의 메모리를 사용하여 연속 처리를 구현한 반면 상기 종래의 연속 처리 구조는 2개의 메모리만을 사용하여 메모리 복잡도를 최소화할 수 있었다.

<47> 그러나, 상기 연속 처리 구조는 Radix-2 알고리즘을 사용함으로 인해 많은 연산 싸이클과 높은 동작 주파수를 요구하는 문제점이 있었다.

【발명이 이루고자 하는 기술적 과제】

<48> 상술한 문제점을 해결하기 위하여 본 발명은 FFT 프로세서에 있어서, 고속 연산과 동시에 최소화된 복잡도를 갖는 회로를 제공함으로써 고성능을 만족시키며 면적을 최소화할 수 있는 프로세서를 제공하는 것을 목적으로 한다.

<49> 상기 목적을 달성하기 위하여 본 발명은 Mixed-radix 알고리즘과 연속처리 구조를 갖는 FFT 프로세서에 있어서, 선택한 메모리와 그 선택한 메모리의 임의의 뱅크로부터 입력과 출력을 수행하기 위한 입출력 인터페이스와; 상기 인터페이스로의 입출력과 FFT 연산에 사용되는 4개의 뱅크로 구성된 2개의 N 워드 메모리와; 상기 선택한 메모리와 뱅크로부터 4개의 버플라이 입력을 공급하기 위한 데이터 교환부와; 상기 데이터 교환부로부터 공급된 Radix-4와 Radix-2 두 종류의 버터플라이를 하나의 회로로 수행하며 대칭적인 리버스 출력을 구성하기 위한 Radix-4/2 버터플라이 연산 회로와; 상기 선택한 메모리와 뱅크로 4개의 버터플라이 출력을 공급하기 위한 데이터 교환부와; 다중 뱅크 메모리 구조에서 인-플레이스 연산을 수행하기 위한 주소를 생성하는 주소 생성부를 포함하는 FFT 프로세서를 특징으로 한다.

【발명의 구성 및 작용】

<50> 이하, 첨부된 도면을 참조로 하여 본 발명을 상세히 설명하기로 한다.

<51> 도4는 본 발명에 따른 인-플레이스 알고리즘을 적용하고 Mixed-radix 구조와 입출력을 동시에 수행함으로써 2N 워드의 메모리로 연속 처리를 수행하는 FFT 프로세서를 나타낸 것으로, 프로세서의 구조는 입출력과 FFT 연산을 위한 메모리 중 입출력을 위한 메모리를 선택하고 메모리의 4개의 뱅크 중 하나의 뱅크를 선택하여 입력과 출력을 수행하기 위한 입출력 인터페이스(101)와; 상기 인터페이스(101)로의 입출력과 FFT 연산이 사용되는 4개의 뱅크로 구성된 2개의 N 워드 메모리(102,103)와; 상기 인터페이스(101)로의 입출력과 FFT 연산을 위한 메모리 중 FFT 연산을 위한 메모리를 선택하고 인-플레이스 연산을 위해 각 버터플라이 입출력에 할당된 뱅크들을 버터플라이 연산 회로의 4개의 입력과 연결하기 위한 제1데이터 교환부(104)와; 상기 제1데이터 교환부(104)에서 공급되는 Radix-4와 Radix-2 두 종류의 버터플라이를 하나의 회로로 수행 가능하며 대칭적인 리버스 출력을 구성하기 위한 버터플라이(105)와; 상기 인터페이스로의 입출력과 FFT 연산을 위한 메모리 중 FFT 연산을 위한 메모리를 선택하고 인-플레이스 연산을 위해 각 버터플라이 입출력에 할당된 뱅크들을 버터플라이 연산 회로의 4개의 출력과 연결하기 위한 제2데이터 교환부(106)와; 상기 다중 뱅크 메모리 구조에서 인-플레이스 연산을 수행하기 위한 뱅크 인덱스 및 주조를 생성하기 위한 주소 생성부(107)로 구성된다.

<52> 상기 FFT 프로세서는 도5와 같은 흐름도를 갖는다. 도5는 32-포인트 Mixed-radix FFT 연산의 예이며 스테이지1과 스테이지2는 Radix-4로, 마지막 스테이지3은 Radix-2로 연산이 수행된다. 본 발명의 구조가 메모리를 4개의 뱅크로 나누어 동시에 4개의 데이터 접근이 가능하므로 2개의 데이터를 사용하는 Raidx-2 버터플라이의 경우 동시에 2개의 버터플라이 연산 수행이 가능하다. 또한 스테이지3의 가는 실선안에 표시한 2개의 Radix-2 버터플라이들이 동시에 수행 가능한 버터플라이 쌍을 나타낸다. 이와 같이 2개의 Radix-2 버터플라이를 동시에 한 싸이클로 수행하면 연산 싸이클 감소의 이득을 볼 수 있다.

<53> 상기 FFT 프로세서에서 Radix-2 버터플라이 구조는 별도의 버터플라이로 구현하지 않고 Radix-4 버터플라이에 데이터 스위칭 회로를 추가하여 구현한다. 이를 도6a에 나타내었다. 도6a에서 'Radix-2'라는 멀티플렉서(Multiplexer) 선택 신호를 통해 Radix-4 버터플라이와 2개의 Radix-2 버터플라이를 구현한다. 도6b는 Radix-4 버터플라이 회로로 Radix-2 버터플라이를 구현하였을 경우 등가의 Radix-2 버터플라이 쌍을 나타낸다.

<54> 다음으로 상기 본 발명의 메모리 구조에서 연속 처리를 수행하기 위한 구조에 대해 설명한다. 종래의 알.래드후앤에 의해 제안된 연속 처리 구조는 Radix-2 알고리즘을 위한 구조인 것에 반해 본 발명의 구조는 Radix-4와 혼합-기수를 위한 구조이다. 또한, 종래의 구조는 FFT와 DIT 연산을 번갈아 가며 수행하는 방식인데 반해 본 발명의 구조는 DIF 연산만을 수행하며 메모리 어드레싱의 제어만으로 연속 처리를 수행한다. 연속 처리를 위해서는 우선 입력과 출력 모두가 순차적 순서로 동시에 수행되어야 한다. 출력된 메모리 위치에 다음 FFT 연산을 위한 입력이 저장되는 방식으로 입출력이 동시에 수행된다. 이는 Radix-4 알고리즘만을 사용할 때 쉽게 구현될 수 있으며 16-포인트 FFT의 예를 도7에 나타내었다.

<55> 도7에서 열1과 열2는 데이터들이 저장되는 메모리 뱅크와 번지를 나타낸 것이고, 열3은 디지트 리버스 출력 순서를 나타낸다. 도7에서 처음으로 외부에서 들어온 입력 데이터들은 열2의 뱅크와 번지에 저정된다. FFT 연산이 수행되면 열3의 디지트 리버스 순서로 출력이 구성되며 메모리 저장 위치는 인-플레이스 연산을 수행하므로 열2의 뱅크와 번지가 유지된다. 연산 결과를 디지트 리버스 어드레싱을 통해 순차적 순서로 출력하고 동시에 순차적 순서로 다음 데이터를 입력하면 열1의 뱅크와 주소에 새로운 입력 데이터들이 저장된다. 열3의 '0'번 출력은 열2의 뱅크0의 0번지에 저장되어 있으므로 이를 출력하고 다음 FFT 연산을 위한 새로운 데이터

의 '0'번 입력을 이 뱅크와 번지에 저장한다. 다음으로 열3의 '1'번 출력은 열2의 뱅크1의 1번지에 있으며 이를 출력하고 새로운 데이터의 '1'번 입력을 이 위치에 저장한다. '2'번 출력은 열2의 뱅크2의 2번지에 저장되어 있으며 이를 출력하고 이 위치에 새로운 데이터의 '2'번 입력을 저장한다. 이와 같은 방법으로 새로운 입력을 저장하면 열1의 뱅크와 번지가 형성된다. 그리고 이에 대한 FFT 연산을 수행한 뒤 순차적 순서로 출력하고 순차적 순서로 입력을 저장하면 다시 열2의 뱅크와 주소 할당으로 복원된다. 따라서 열1과 열2의 뱅크와 주소 할당이 번갈아 가며 수행된다.

<56> 이와 같이 순차적 순서로 동시에 입출력 수행이 가능하면 하나의 메모리가 연산을 수행하는 동안 나머지 메모리는 입출력을 수행하는 방식으로 메모리 2개만으로 연속 처리가 가능하다. 이때 FFT 연산은 입출력 동작 주파수에 비해 2배의 동작 주파수로 연산이 수행되어야 한다. 이는 상기 표1에서 볼 수 있듯이 Radix-4의 경우 1024-포인트 FFT부터, Radix-4/Radix-2의 혼합-기수의 경우 512-포인트 FFT부터 FFT 길이 N보다 연산 클럭 싸이클이 더 크기 때문이다.

<57> 혼합-기수의 경우 연속 처리를 위한 순차적 입출력을 위해 별도의 조작이 필요하다. 32-포인트 혼합-기수의 경우 출력이 도5의 열4와 같이 나타난다. 이는 도7 열3의 Radix-4 알고리즘의 리버스 순서와 달리 비대칭적인 리버스 형태를 갖는다. 먼저 Radix-4 알고리즘만을 사용한 경우의 디지트 리버스 순서에 대해 설명하면, Radix-4의 2^n-포인트 FFT에 대한 디지트 리버스 순서를 도8a에 나타내었다. 2^n개의 데이터를 카운트하기 위해서 n 비트가 필요하므로 n 비트 카운터가 사용된다. 도8a에서 (n-1, n-2) 번째 비트, (n-3, n-4) 번째 비트, …, (3, 2) 번

째 비트, (1, 0) 번째 비트들이 디지트를 이루어 리버스가 수행된다. 도8a에서 알 수 있듯이 디지트들의 정가운데를 중심으로 대칭적으로 리버스가 됨을 알 수 있다.

<58> 도8b는 2^n-포인트 FFT에 대한 혼합-기수의 리버스 순서를 타나내었다. 혼합-기수의 경우 23, 25, 27, 29 등의 2의 홀수승의 포인트 수를 가져 출력 카운트 비트의 수가3, 5, 7, 9 비트 등과 같이 홀수이기 때문에 Radix-4와 같이 2 비트 디지트로만 리버스할 수 없다. 최하위 비트가 별도로 리버스 되어야 하며 이로 인해 비대칭적인 리버스 형태를 갖게 된다. 도8b의 예로 32(2^5)-포인트에 해당하는 도5의 열4는 도9와 같은 비대칭적 리버스 형태를 가진다. 비대칭적인 리버스 출력을 가질 경우 Radix-4 알고리즘으로 구성된 도7과 같이 열1과 열2의 뱅크와 번지가 반복되는 구조를 구성할 수 없다. 혼합-기수에서도 Radix-4만을 사용하는 도7과 같이 연속 처리를 위해 열1과 열2가 반복되는 구조를 가지려면 출력이 대칭적인 리버스 형태를 가져야 한다. 이를 위해 혼합-기수 알고리즘에서 도5의 열4와 같은 비대칭적 출력 순서를 열3과 같은 대칭적 출력 순서를 갖도록 변환하는 데이터 교환이 수행되며 이를 통한 32(2^5)-포인트의 대칭적 리버스 출력 순서가 도5의 열3과 같으며 도10과 같이 생성할 수 있다. 이를 일반화하여 2^n-포인트 혼합-기수 FFT의 경우를 살펴보면 출력의 대칭적 리버스 순서는 도11과 같으며 상위 2 비트(n-1, n-2)와 하위 2비트 (1, 0)는 디지트 리버스가 수행되며 가운데 비트들(n-3, n-4, … , 3, 2)은 비트 리버스가 수행된다. 결론적으로 원래 도8b의 비대칭적 리버스 형태가 데이터 교환 과정을 통해 도11의 대칭적 리버스 형태로 변환되는 것이다.

<59> 도5의 열3과 도10에 나타낸 32-포인트의 대칭적 리버스 순서를 구현하기 위한 데이터 교환 과정을 도12에 나타내었다. 도12는 도5의 우측 상단에 굵은 실선으로 표시한 8-포인트 DFT 부분에 해당한다. 도12에 나타낸 바와 같이 단순히 스테이지2에서는 Radix-4 버터플라이의 두 번째와 세 번째 출력 g'2(n), g'3(n)의 저장 위치를 교환하고 스테이지의 Radix-2 버터플라이

쌍에서도 X'2(n)과 X'3(n)의 출력 저장 위치를 교환하면 도12의 열2의 비대칭적 리버스 순서가 열1의 대칭적 리버스 순서로 변환된다. 데이터 교환은 도6a의 버터플라이 회로에서 'Exchange' 신호의 제어를 통해 수행될 수 있다. 예로 든 32-포인트 FFT 뿐만 아니라 모든 2^n(n=1, 3, 5, 7, 9, …)-포인트 FFT에서 첫번째 스테이지를 제외한 나머지 스테이지에서 Radix-4 버터플라이의 두번째와 세번째 출력 저장 위치를 교환하고 마지막 스테이지에서 2개의 Radix-2 버터플라이의 두번째와 세번째 출력 저장 위치를 교환하는 것에 의해 대칭적 리버스 출력 순서를 구성하는 것이 가능하다.

<60> 마지막으로 Mixed-radix 알고리즘에서의 뱅크 인덱스 생성에 대해 설명한다. 이 뱅크 인덱스 생성 방법에 의해 도5의 열1과 열2에 나타낸 뱅크 인덱스가 생성된다. 뱅크 인덱스와 각 뱅크의 번지 생성은 FFT 길이가 2^n일 경우 n 비트 카운더를 이용하여 생성한다. Radix-4 알고리즘의 경우 (2^2, 2^4, 2^6, 2^8 등의 포인트 수) 뱅크 인덱스(뱅크i) 생성은 2비트 디지트들의 모듈로-4 덧셈으로 수행된다. 그러나, Mixed-radix의 경우 2^3, 2^5, 2^7, 2^9 등과 같은 포인트 수를 가져 입력 카운트 비트의 개수가 3비트, 5비트, 7비트, 9비트 등과 같이 홀수이므로 2 비트 디지트들의 모듈로-4 덧셈만으로는 뱅크 인덱스를 생성할 수 없다. Mixed-radix의 경우 2비트 디지트들로 입력 카운트 비트들을 나누고 남은 1비트를 모듈로-4 덧셈에 포함하여야 한다. 32(2^5)-포인트 FFT의 2비트 디지트들과 1비트를 구성하는 방법을 도13에 타나내었다. 이때 별도의 1비트는 입력 데이터 카운트 비트 중 3번 비트에 해당하며 2비트 디지트들은 (4, 2), (1, 0)에 해당한다. 이와 같이 뱅크를 생성하면 FFT 연산시 모두 다른 뱅크로부터 데이터를 가져올 수 있을 뿐만 아니라 도5의 열2에서 열1로 변경시 뱅크 인덱스 순서는 그대로 유지할 수 있다

.

<61> 도 14는 32-포인트에서의 뱅크 인덱스 생성 방법을 일반화하여 2n-포인트 Mixed-radix FFT 일 때의 뱅크 인덱스 생성 방법을 나타낸 것이다. 도14에서 입력 데이터 카운트 비트들 중 별도의 1비트 위치는 n-2번째 비트이며 2비트 디지트들의 모듈로-4 덧셈에 이 별도의 비트가 포함된다.

【발명의 효과】

<62> 본 발명은 상술한 바와 같이 Radix-4 알고리즘에 기초한 Mixed-radix 알고리즘을 사용하여 고속 연산이 가능하고, Mixed-radix 알고리즘에 인-플레이스 연산을 적용하고 입출력 동시 수행을 통해 4개의 뱅크로 구성된 N 워드 메모리 2개로 연속 처리를 수행함으로 인해 사용되는 면적을 최소화 할 수 있다.

<63> 종래의 FFT 프로세서와 본 발명의 FFT 프로세서의 연산 싸이클 비교를 표3에 나타내었다. 표3으로부터 Radix-2 알고리즘을 사용한 종래의 메모리 구조에 비해 연산 싸이클이 약 1/4로 감소됨을 알 수 있다.

<64> 【표 3】

Radix-2 인-플레이스 FFT 구조와의 비교

구 조	알고리즘 클럭 싸이클	N = 2,048	N = 4,096
종래의 메모리 구조	Radix-2 $\frac{N}{2}$ log2N+2	11,266	24,578
본 발명의 구조	Radix-4 $\frac{N}{4}$ log4N+6	-	6,150
	Mixed-radix (Radix-4, Radix-2) $\frac{N}{4}$ log42N+6	3,078	-

<65> 또한, 다중 뱅크 구조를 사용하지 않은 Radix-4 알고리즘을 기반으로 한 종래의 Mixed-radix FFT 프로세서와 비교할 경우 본 발명의 구조에 비해 약 4배 가까운 연산 싸이클이 소모된다.

<66> 인-플레이스 알고리즘과 동시에 입출력하는 구조를 채택하지 않은 Mixed-radix FFT 프로세서는 4개의 뱅크로 구성된 N 워드 메모리 4개를 요구하여 N 워드 메모리 2개를 요구하는 본 발명의 구조에 비해 2배의 메모리 면적을 가진다.

<67> 따라서, 본 발명은 고속 연산뿐만 아니라 적은 하드웨어 크기도 만족하여 OFDM, DMT 시스템에 적용하기 용이하다.

【특허청구범위】

【청구항 1】

입출력과 FFT 연산을 위한 메모리 중 입출력을 위한 메모리를 선택하고 메모리의 4개의 뱅크 중 하나의 뱅크를 선택하여 입력과 출력을 수행하기 위한 입출력 인터페이스와;

상기 인터페이스로의 입출력과 FFT 연산이 사용되는 4개의 뱅크로 구성된 2개의 N 워드 메모리와;

상기 인터페이스로의 입출력과 FFT 연산을 위한 메모리 중 FFT 연산을 위한 메모리를 선택하고 인-플레이스 연산을 위해 각 버터플라이 입출력에 할당된 뱅크들을 버터플라이 연산 회로의 4개의 입력과 연결하기 위한 제1데이터 교환부와;

상기 제1데이터 교환부에서 공급되는 Radix-4와 Radix-2 두 종류의 버터플라이를 하나의 회로로 수행 가능하며 대칭적인 리버스 출력을 구성하기 위한 버터플라이와;

상기 인터페이스로의 입출력과 FFT 연산을 위한 메모리 중 FFT 연산을 위한 메모리를 선택하고 인-플레이스 연산을 위해 각 버터플라이 입출력에 할당된 뱅크들을 버터플라이 연산 회로의 4개의 출력과 연결하기 위한 제2데이터 교환부와;

상기 다중 뱅크 메모리 구조에서 인-플레이스 연산을 수행하기 위한 뱅크 인덱스 및 주소를 생성하기 위한 주소 생성부를 포함하는 것을 특징으로 하는 연속 처리 구조를 갖는 인-플레이스 알고리즘 혼합-기수 FFT 프로세서.

【청구항 2】

제 1 항에 있어서,

상기 버터플라이는 Radix-4 버터플라이와 2개의 Radix-2 버터플라이의 출력을 교환하는 것에 의해 대칭적 리버스 출력 순서를 구성하여 순차적 입출력을 동시에 수행 가능하게 함으로써 4개의 뱅크로 구성된 N 워드 메모리 2개만으로 연속 처리를 수행할 수 있는 Radix-4/2 버터플라이 연산회로인 것을 특징으로 하는 연속 처리 구조를 갖는 인-플레이스 알고리즘 혼합-기수 FFT 프로세서.

【청구항 3】

제 1 항에 있어서,

상기 인-플레이스 알고리즘은 Radix-4 알고리즘의 다중 뱅크 메모리 구조의인-플레이스 알고리즘을 변형하여 혼합-기수 알고리즘을 위한 인-플레이스 알고리즘인 것을 특징으로 하는 연속 처리 구조를 갖는 인-플레이스 알고리즘 혼합-기수 FFT 프로세서.

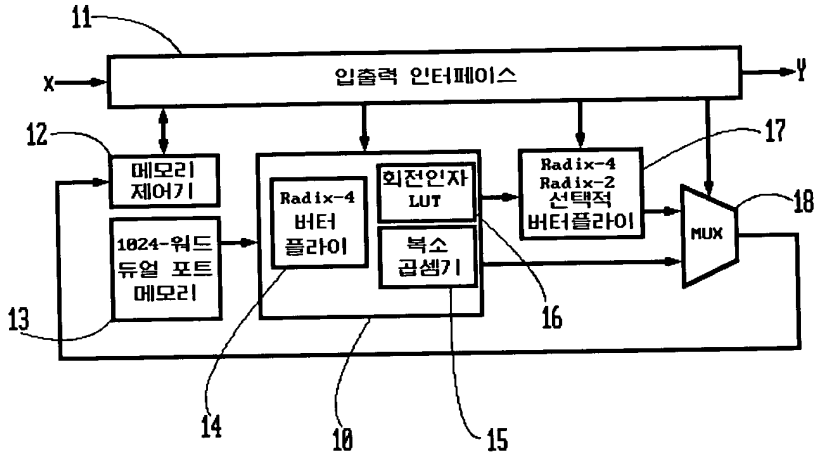
【청구항 4】

제 1 항에 있어서,

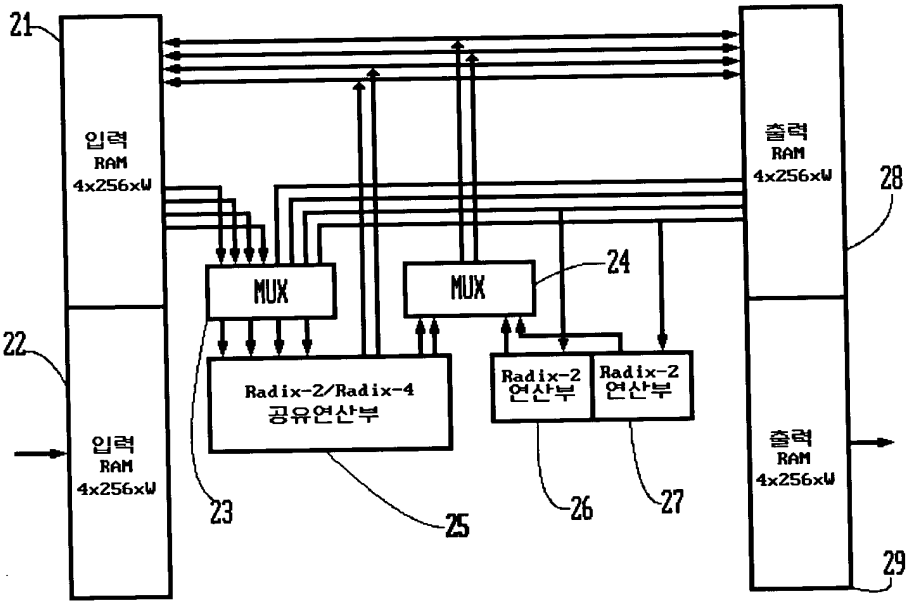
상기 인-플레이스 알고리즘은 뱅크 인덱스 생성에서 2^n-포인트 연산일 때 n-2번째 비트를 별도로 모듈로-4 덧셈에 포함하는 것을 특징으로 하는 연속 처리 구조를 갖는 인-플레이스 알고리즘 혼합-기수 FFT 프로세서.

【도면】

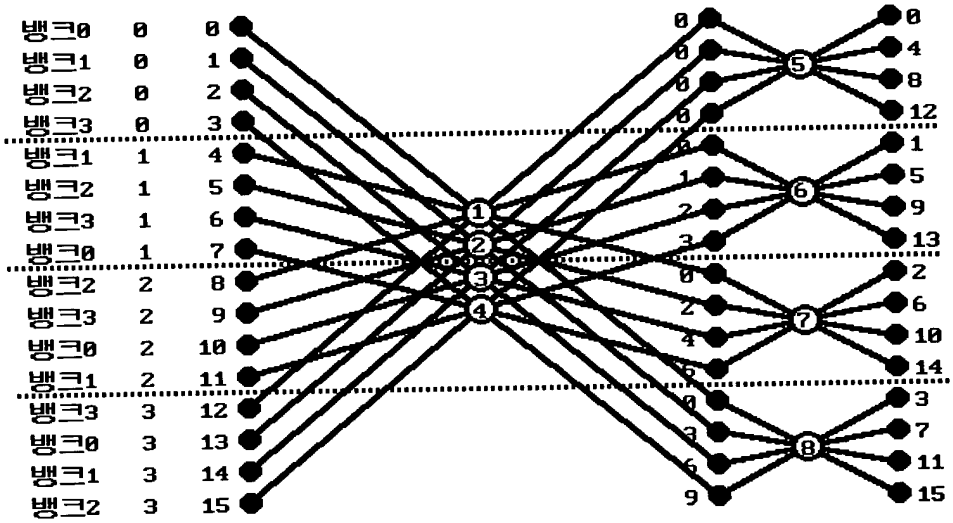
【도 1】


11
입출력 인터페이스
X
Y
12
메모리 제어기
1024-워드 듀얼 포트 메모리
13
Radix-4 버터 플라이
회전인자 LUT
복소 곱셈기
14
10
15
16
Radix-4 Radix-2 선택적 버터플라이
17
MUX
18

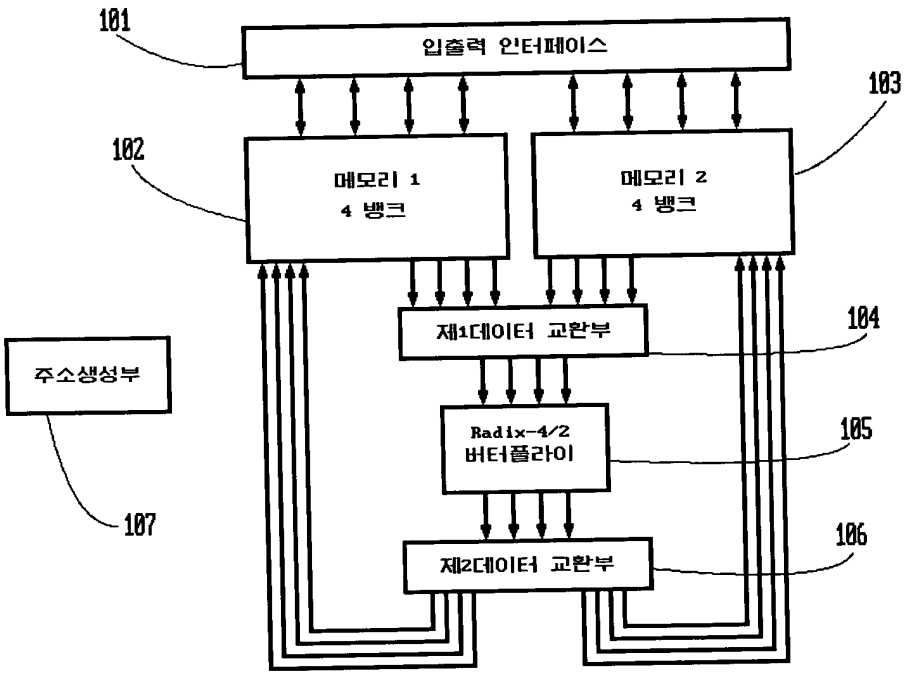
【도 2】


21
입력 RAM 4x256xW
22
입력 RAM 4x256xW
MUX
MUX
24
Radix-2/Radix-4 공유연산부
Radix-2 연산부
Radix-2 연산부
23
25
26
27
출력 RAM 4x256xW
28
출력 RAM 4x256xW
29

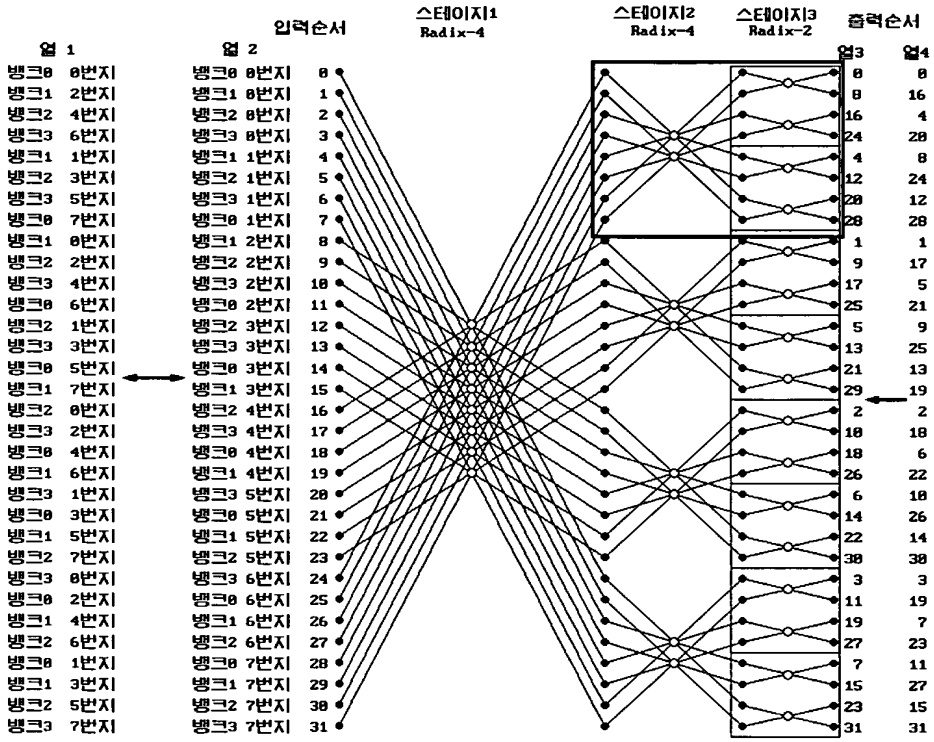
【도 3】


뱅크0 0 0
뱅크1 0 1
뱅크2 0 2
뱅크3 0 3
뱅크1 1 4
뱅크2 1 5
뱅크3 1 6
뱅크0 1 7
뱅크2 2 8
뱅크3 2 9
뱅크0 2 10
뱅크1 2 11
뱅크3 3 12
뱅크0 3 13
뱅크1 3 14
뱅크2 3 15
0 4 8 12
1 5 9 13
2 6 10 14
3 7 11 15

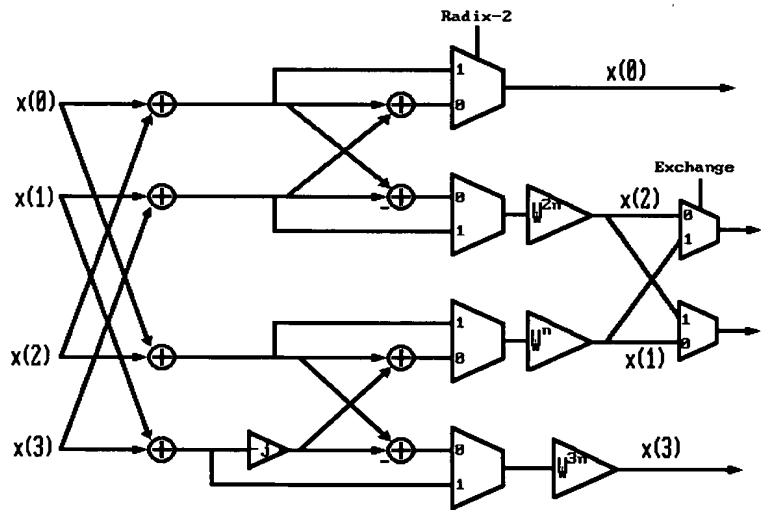
【도 4】


101
입출력 인터페이스
103
102
메모리 1
4 뱅크
메모리 2
4 뱅크
104
제1데이터 교환부
주소생성부
Radix-4/2
버터플라이
105
107
106
제2데이터 교환부

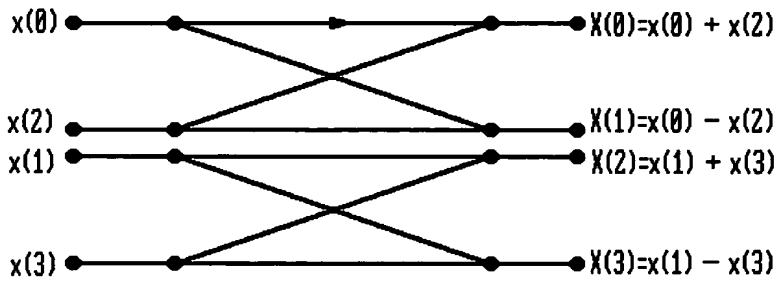
【도 5】


입력순서
스테이지1
Radix-4
스테이지2
Radix-4
스테이지3
Radix-2
출력순서
열 1
열 2
열3
열4
뱅크0 0번지 0
뱅크1 2번지 1
뱅크2 4번지 2
뱅크3 6번지 3
뱅크1 1번지 4
뱅크2 3번지 5
뱅크3 5번지 6
뱅크0 7번지 7
뱅크1 0번지 8
뱅크2 2번지 9
뱅크3 4번지 10
뱅크0 6번지 11
뱅크2 1번지 12
뱅크3 3번지 13
뱅크0 5번지 14
뱅크1 7번지 15
뱅크2 0번지 16
뱅크3 2번지 17
뱅크0 4번지 18
뱅크1 6번지 19
뱅크3 1번지 20
뱅크0 3번지 21
뱅크1 5번지 22
뱅크2 7번지 23
뱅크3 0번지 24
뱅크0 2번지 25
뱅크1 4번지 26
뱅크2 6번지 27
뱅크0 1번지 28
뱅크1 3번지 29
뱅크2 5번지 30
뱅크3 7번지 31
0 0
8 16
16 4
24 20
4 8
12 24
20 12
28 28
1 1
9 17
17 5
25 21
5 9
13 25
21 13
29 19
2 2
10 18
18 6
26 22
6 10
14 26
22 14
30 30
3 3
11 19
19 7
27 23
7 11
15 27
23 15
31 31

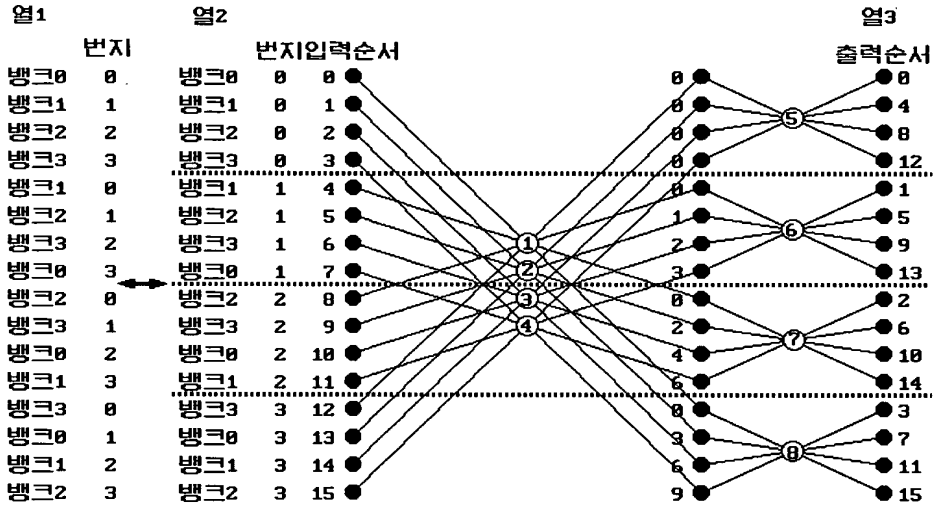
【도 6a】


Radix-2
x(0)
x(1)
x(2)
x(3)
Exchange
x(0)
x(2)
x(1)
x(3)
1
0
-1
W^{2n}
W^{n}
W^{3n}

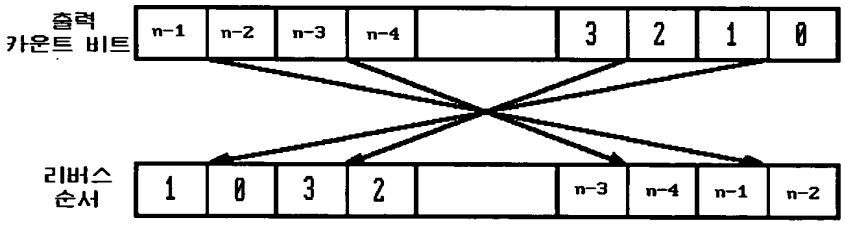
【도 6b】


x(0)
x(2)
x(1)
x(3)
X(0)=x(0) + x(2)
X(1)=x(0) - x(2)
X(2)=x(1) + x(3)
X(3)=x(1) - x(3)

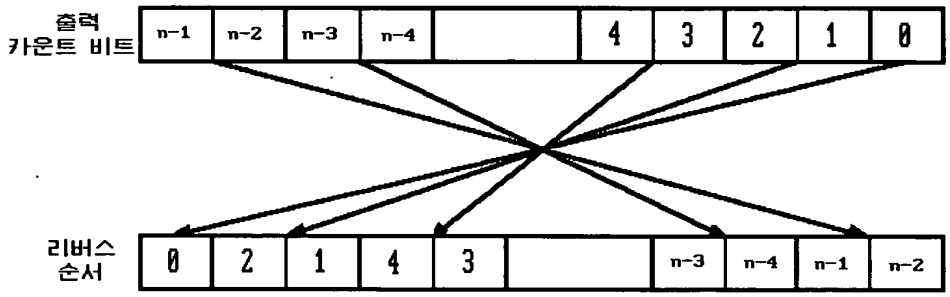
【도 7】


열1
번지
열2
번지입력순서
열3
출력순서

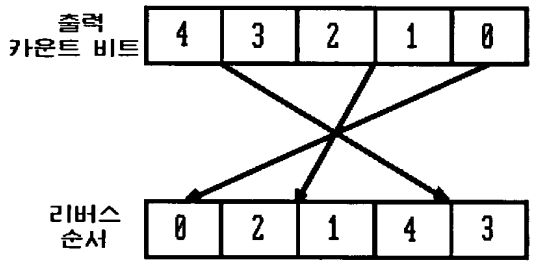
【도 8a】


출력
카운트 비트
n-1
n-2
n-3
n-4
3
2
1
0
리버스
순서
1
0
3
2
n-3
n-4
n-1
n-2

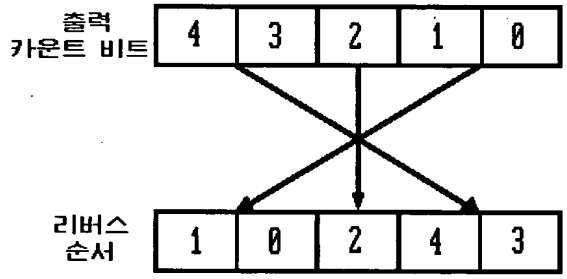
【도 8b】



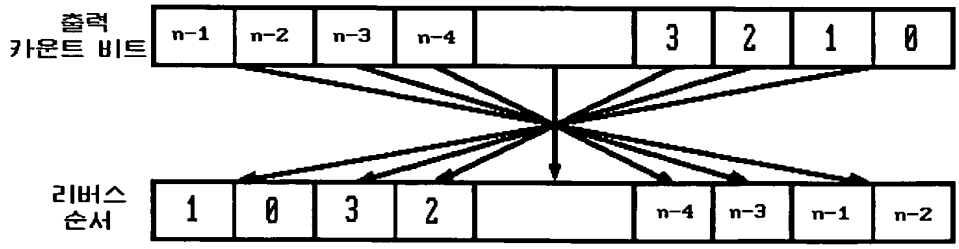
【도 9】



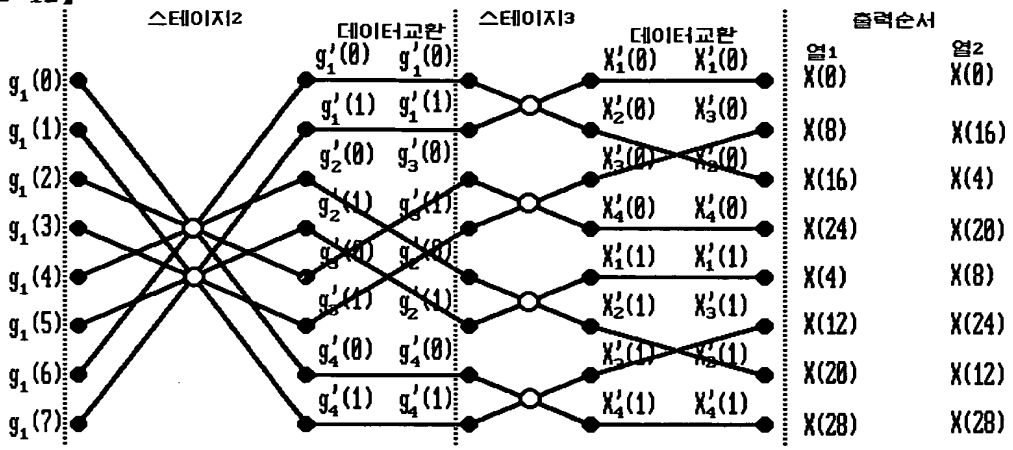
【도 10】



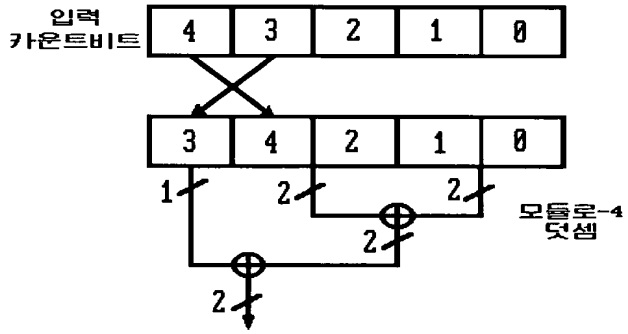
【도 11】



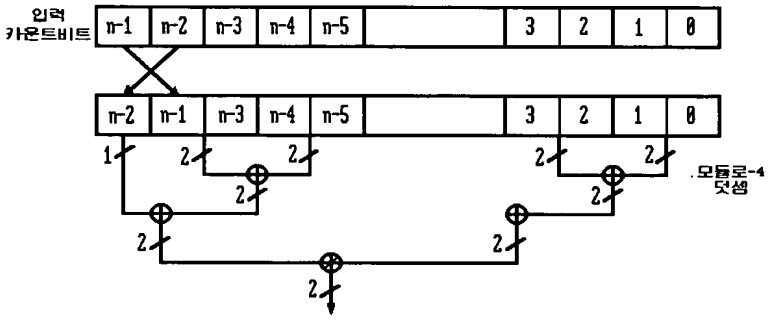
【도 12】


스테이지2
데이터교환
스테이지3
데이터교환
출력순서
열1
열2
X(0) X(0)
X(8) X(16)
X(16) X(4)
X(24) X(20)
X(4) X(8)
X(12) X(24)
X(20) X(12)
X(28) X(28)

【도 13】


입력
카운트비트
4 3 2 1 0
3 4 2 1 0
모듈로-4
덧셈

【도 14】


입력
카운트비트
n-1 n-2 n-3 n-4 n-5 3 2 1 0
n-2 n-1 n-3 n-4 n-5 3 2 1 0
모듈로-4
덧셈